This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

COALA: Concurrently Assigning Wire Segments to Layers for 2D Global Routing

Yun-Jhe Jiang and Shao-Yun Fang, *Member, IEEE*

***Abstract*—Two-dimensional (2D) global routing followed by layer assignment is a common and popular strategy to obtain a good trade-off between runtime and routing performance. Yet, the huge gap between 2D routing patterns and the final 3D routing paths often results in inevitable overflow after layer assignment. State-of-the-art studies on layer assignment usually adopt dynamic programming-based approaches to sequentially find an optimal solution for each net in terms of overflow or/and the number of vias. However, a fixed assignment ordering severely restricts the solution space, and the distributed overflows can hardly be resolved with any existing refinement approach. This paper proposes a novel layer assignment framework that concurrently considers all the wire segments of nets and iteratively assigns them from the lowest available layer to the highest one. The concurrent scheme facilitates the maximal utilization of routing resource on each layer, contributing to an effective re-routing procedure that greatly reduces inevitable overflows. Based on the proposed framework, we further propose an obstacle-aware strategy that can mitigate obstacle-induced inevitable overflows in the original framework. Experimental results show that compared to an implemented sequential layer assignment approach based on state-of-the-art (SOTA) techniques and refined by the well-known overflow/congestion reduction rip up and re-routing procedure, the proposed COALA framework brings great improvements in the overflow reduction and runtime efficiency, which shows the significant advantage of the concurrent layer assignment scheme over sequential methods. The improvement is also verified in detailed routing, where the proposed COALA framework contributes to sparser routing results with fewer vias and design rule violations (DRVs).**

I. Introduction

Due to the increasing complexity of designs and decreasing sizes of circuit devices, limited routing resource greatly complicates the cell routing problem. A global router plays a critical role in the back-end design flow, which partitions routing layers into global tiles, plans a global routing path for each net, and minimizes the total overflow exceeding tile capacities. To refine the congestion regions and overflows, 3D rip-up and rerouting is adopted in almost all the global routers and spends most of the runtime in the whole global routing process [13], [6]. Therefore, some routers adopt multi-level approaches to reduce the runtime in global routing [13], [22].

There are two main strategies to solve the global routing problem: three-dimensional (3D) global routing and two-dimensional (2D) global routing followed by layer assignment. A 3D global router transforms the global tiles of all routing layers into a 3D global routing graph and applies maze routing or an integer linear programming (ILP) formulation to directly derive 3D routing paths [18], [21]. Due to the increasing number of routing layers and chip size, the extremely high routing complexity can result in prohibitively large runtime cost.

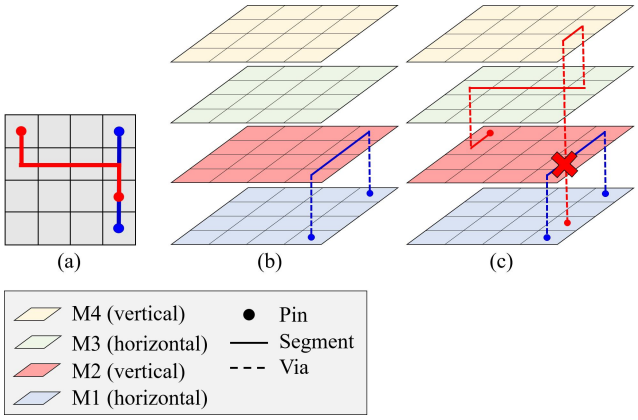

Fig. 1: A layer assignment example with a conventional DP-based layer assignment method. (a) The given 2D global routing routing paths. (b) The layer assignment result of the blue net if it is assigned first. (c) Since the red net is assigned after the blue net, its optimal solution still suffers from an overflow.

To obtain an acceptable trade-off between runtime and routing performance, most of the global routers in literature adopt the second strategy [1], [2], [12], [16], [23]. They first compress the 3D routing layers into a 2D routing graph and compute the capacity of each 2D global tile. Then, a multi-pin net could be decomposed into two-pin nets with a Steiner tree-based method, and each two-pin net can be routed with an L-route, Z-route, or monotonic route for initial routing [11]. The initial routing may cause some congested regions with large 2D estimated overflow, and thus the nets crossing those regions could be ripped-up and rerouted [16].

One of the biggest challenges of 2D global routing in combination with layer assignment is due to the huge gap between 2D routing patterns and the final 3D routing paths. During 2D routing, the only available information is how many paths pass through a global tile and how many paths turn at the tile. However, the exact number of required vias among layers can only be obtained after layer assignment, which may cause more overflow that is unexpected in 2D global routing, resulting in large effort in detailed routing and rip-up and rerouting. To tackle via demands of global routing paths, existing studies have proposed solutions in several aspects. The first type of strategies is to additionally consider via demands

This work was partially supported by TSMC, Synopsys, Siemens EDA, and MOST of Taiwan under Grant No's MOST 110-2223-E-011-002-MY3. A preliminary version was published at IEEE/ACM International Conference on Computer-Aided Design 2020 [5].

Yun-Jhe Jiang and Shao-Yun Fang are with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: {d10907011, syfang}@mail.ntust.edu.tw).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

in 2D global routing models. [17], [20] consider various design rules of vias, [19] resolves local nets with non-uniform g-cells, and [4] considers in-tile wires and remaining via capacity in their models. However, the estimated via demand in the global routing stage suffers from low accuracy due to unpredictable layer assignment behavior. The second type of strategies is to consider the actual via demands during layer assignment. [8] is the first work simultaneously considering wire and via demands with detailed layer assignment steps, and [3] additionally divides large nets into small sub-nets to improve solution quality. In addition, existing studies also resort to various refinement techniques for layer assignment to reduce overflow, such as layer shifting [3], edge shifting [15], and rip-up and reassignment [3], [10], [20].

Although minimizing via usage and considering via capacity during layer assignment is effective in controlling via overflow, existing layer assignment approaches suffer from two common drawbacks. First, most of the above works sequentially performs layer assignment for each net based on dynamic programming (DP)-based algorithms. In spite of the optimality of a DP-based method that minimizes the overflow increment and the number of vias for each net, the assignment ordering of all nets severely restricts the solution space, making the overall assignment result far from optimal. Second, the DP-based approaches cause difficulties in the assignment refinement process. For a tile with large overflow, deciding or iteratively trying which segments should be ripped up and reassigned/shifted critically determines the final solution quality and becomes another complicated optimization problem. In addition, the resulting overflows are randomly scattered on segments, and thus the existing refinement techniques are only performed on each individual wire segment suffering from overflow, limiting the effectiveness in overflow reduction.

Fig. 1 illustrates a non-ideal layer assignment result due to the sequential layer assignment strategy. In this example as well as the benchmarks we use in the experiments, wires can only route horizontally on odd layers and route vertically on even layers, and M1 is only for pins and not available for layer assignment. The edge capacity of a global tile on every layer in Fig. 1 is set to be one, and there are two available vertical layers and one available horizontal layer. Fig. 1(a) shows two two-pin nets with their feasible 2D global routing paths. If the blue net is routed first in a sequential layer assignment approach, as shown in Fig. 1(b), the vertical wire segment is more likely to be assigned on M2 to minimize the number of vias. Then, the red net will be assigned as that in Fig. 1(c). Although the assignment of the red net is the optimal solution based on the already assigned result of the blue net, it incurs an overflow on M2. This overflow can be resolved if the ordering of the blue net and the red net is reversed, while an optimal ordering can hardly be found by using simple heuristics adopted by the above existing works. In addition, the best layer assignment result may not be achieved by any ordering with a sequential assignment procedure, which is also a well-known limitation in sequential routing algorithms.

Note that there exist some studies proposing Lagrangian relaxation or integer programming-based approaches to consider the layer assignments of multiple nets such as [9] and [14]. However, these approaches just perform segment reassignment based on an initial layer assignment solution, and only partial nets can be considered at a time due to the huge problem complexity, which may implies that it is impractical to simultaneously perform layer assignments for all nets using mathematical formulations.

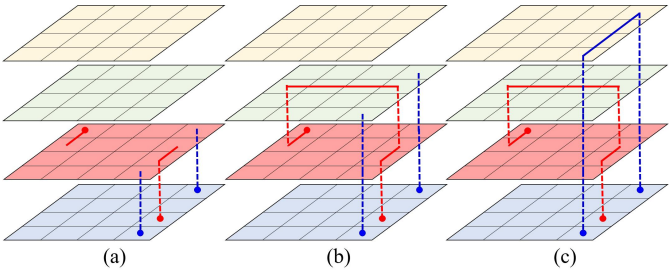

Fig. 2: The same layer assignment instance with our proposed concurrent layer assignment method. (a) The wire segment assignment for M2 and the via assignment for VIA12. (b) The wire segment assignment for M3 and the via assignment for VIA23. (c) The wire segment assignment for M4 and the via assignment for VIA34.

In this paper, we propose a novel layer assignment framework that can concurrently consider the wire segments of all nets and assign them from the lowest to the highest layers, which mitigates the net ordering problem and takes into account via demands at the same time. The concept of our idea is illustrated in Fig. 2. For the same layer assignment instance as that in Fig. 1, we first focus on the assignment of wire segments on the lowest available layer, M2 in this example, by considering the vias of all nets between M1 and M2, as shown in Fig. 2(a). Similar process is then iteratively performed for higher layers, as illustrated in Figs. 2(b) and (c), where a layer assignment solution without overflow can be found. The major contributions of this paper are listed as follows:

- This paper is, to the best of our knowledge, the first work presenting a concurrent layer assignment framework (COALA), which mitigates the net ordering problem that previous studies suffer from, and assigns wire segments of all nets layer by layer to minimize the total overflow.
- A 3D demand prediction map is proposed to calculate the least demand each tile of the current layer should be assigned. The prediction map can guide the wire segment assignment process to fully utilize the routing resource and minimize overflows on the topmost layer.
- Three novel techniques for the concurrent scheme are proposed. The first technique assigns complete segments as many as possible. The second technique fragments wire segments to maximize resource utilization and to avoid inevitable overflow by using the demand prediction map. After concurrent layer assignment, the last technique finds other global routing paths for the sub-nets that were not assigned onto layers due to overflow by using 3D endpoint re-routing, which is greatly different from the conventional rip-up and re-routing process used after a sequential layer assignment approach.
- An obstacle-aware strategy is proposed to reduce obstacle-induced inevitable overflows in the basic COALA flow.
- Experimental results show that compared to the DP-based methods, the proposed COALA framework can averagely reduce the maximum overflow by 32%/20% and reduce the number of tiles with overflows by 33%/9% with much less runtime for the 2018/2019 ISPD Contest benchmarks.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

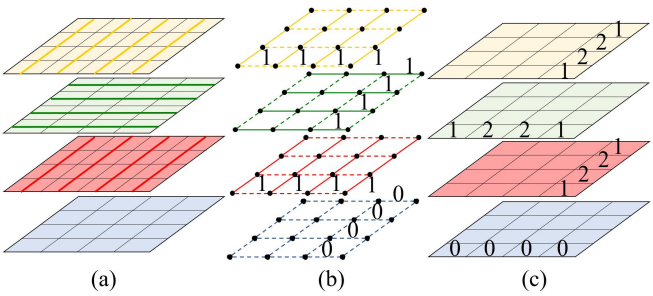

Fig. 3: The generation of the adopted capacity map. (a) The tiles and tracks on each layer. (b) The edge capacities in the global routing graph. (c) The adopted 3D vertex capacity map.

- Based on the same state-of-the-art (SOTA) detailed router TritonRoute, compared to the DP-based methods, the proposed COALA framework can generate sparser routing results with fewer DRVs and vias.

The remaining of the paper is organized as follow: Section II introduces the proposed 3D capacity and demand prediction maps. Then, Section III details each step in the basic COALA flow. After that, Section IV introduces the obstacle-aware strategy. Finally, Section V presents the experimental results, and Section VI gives the conclusions.

II. Capacity and Demand Prediction Maps

In this section, the adopted capacity map and the proposed demand prediction map facilitating the concurrent layer assignment process are introduced.

A. 3D Vertex Capacity Map

In this work, we calculate the capacity of a tile on each layer to limit the numbers of wire segments and vias assigned to the tile. In a conventional 2D capacity map for global routing, the capacity is assigned on the boundary of two adjacent global tiles and is calculated as the number of passing tracks. Fig. 3(a) shows the 3D layout planes with routing tracks, which is the same as that in Fig. 1. Since only one track is available in this example, the 3D global routing graph with edge capacities is show in Fig. 3(b). Since these edge capacities cannot take the via demand in each tile into account, we adopt a 3D vertex capacity map. The capacity of a tile $T(i,j,k)$ at Layer i, Row j, and Column k is calculated as follows:

$$Cap(i,j,k) = \begin{cases} G_l(i,j,k) + G_r(i,j,k), \\ \qquad \text{if Layer } i \text{ is horizontal.} \\ G_t(i,j,k) + G_b(i,j,k), \\ \qquad \text{if Layer } i \text{ is vertical.} \end{cases} \quad (1)$$

where $Cap(i,j,k)$ represents the capacity of $T(i,j,k)$, $G_l(i,j,k)$/$G_r(i,j,k)$ is the edge capacity of the left/right boundary of $T(i,j,k)$ if Layer i is a horizontal layer, and $G_t(i,j,k)$/$G_b(i,j,k)$ is the edge capacity of the top/bottom boundary of $T(i,j,k)$ if Layer i is a vertical layer. Therefore, the capacity of each tile vertex is computed by summing the capacities of the two adjacent edges. Fig. 3(c) shows a row/column of vertex capacities of a horizontal/vertical layer. Note that since we assume that M1 is congested and leaves not much routing resource, the capacities are zero for the tiles on M1.

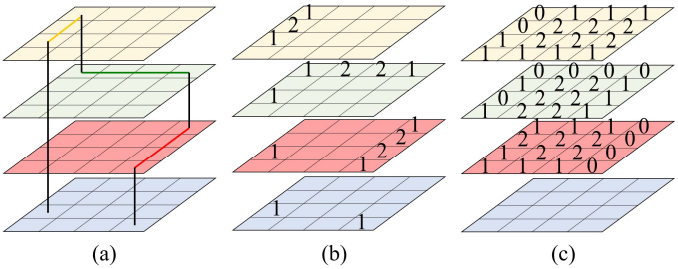

Fig. 4: An example of demand calculation. (a) A global guide with segments on M2, M3 and M4. (b) The demand of the global guide in Fig. 4(a). (c) The remained capacities.

To consider the demands of wire segments and vias in a similar way, if a wire segment is assigned to the tiles $T(i,j,k_1), T(i,j,k_2), \ldots, T(i,j,k_n)$ on a horizontal layer, the segment costs two units of demand for $T(i,j,k_2), ..., T(i,j,k_{n-1})$ because it passes both the two boundaries for each of them, and the segment costs one unit of demand for $T(i,j,k_1)$ and $T(i,j,k_n)$. In addition, we assume that a stacked via costs one unit of demand, and a local wire (all the pins of a net are in the same 3D global tile, $T(i,j,k)$) costs one unit of demand respectively on $T(i,j,k)$ and $T(i+1,j,k)$ in our model. Fig. 4 is used to explain the demand calculation. Fig. 4(a) shows a global guide passing through different layers, and the corresponding demand is given in Fig. 4(b). The remaining vertex capacities shown in Fig. 4(c) are derived by subtracting the demands in Fig. 4(b) from the original vertex capacities in Fig. 3(c). The overflow of a tile $T(i,j,k)$ can be calculated as follows:

$$OV(i,j,k) = \max(Dem(i,j,k) - Cap(i,j,k), 0), \quad (2)$$

where $OV(i,j,k)$ and $Dem(i,j,k)$ respectively represent the number of overflows and demands in $T(i,j,k)$. The calculation indicates that when the demand is relative larger than the capacity of a tile, the overflow occurs.

Note that we use these simple capacity and demand models in this paper for better explanation in the following steps. More complicated models that may distinguish segment capacity and via capacity) in existing studies [4], [7], [8] or developed in the future can be trivially embedded into the proposed flow for further application.

B. Demand Prediction Map

In addition to the capacity map, we propose a novel demand prediction map to effectively reduce the resulting overflow. The idea is to infer the minimum wire segments needing to be assigned to a currently being processed layer by considering the capacities of its upper layers.

Fig. 5 shows the process of prediction map initialization in the rightmost column of the example that we consistently used in the previous figures. Fig. 5(a) shows the partial capacity map composed of the rightmost columns on M2 and M4. Fig. 5(b) shows the two vertical segments lying in the rightmost column of Fig. 1(a), and the right numbers indicate the 2D demands of the column. The values in the prediction map are determined from the highest layer to the lowest one. Each value indicate the maximum demand each tile can accommodate. As shown in Fig. 5(c), the total demands of the four 2D tiles in the column is $\{1,3,3,1\}$, and the four tiles on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

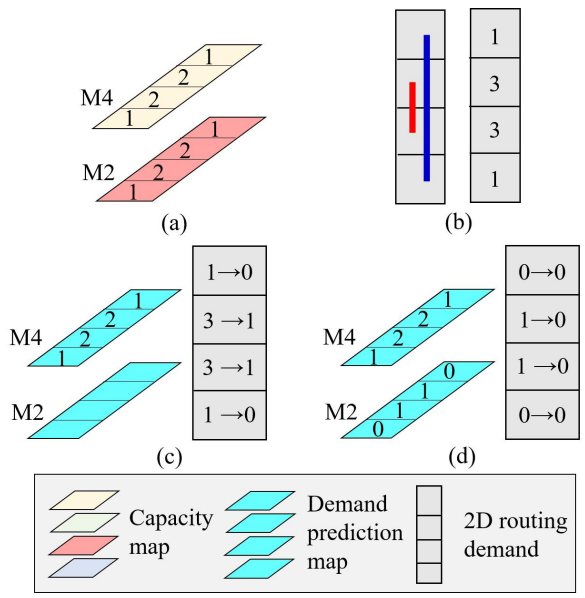

Fig. 5: The calculation of the demand prediction map. (a) The 3D capacities in the rightmost column of Fig. 1(a). (b) The 2D demands of the two segments. (c) The predicted demand values on M4. (d) The predicted demand values on M2.

M4 can separately accommodate $\{1, 2, 2, 1\}$ of them according to the capacity map. After assigning the demands to M4, the remaining demands of the column become $\{0, 1, 1, 0\}$. These remaining demands can be accommodated in M2, as illustrated in Fig. 5(d).

We can observe from Fig. 5(d) that the demand values in the prediction map on M2 imply the minimum demands each tile needs to be assigned when layer assignment is performed on M2; otherwise, some overflow must occur in this column, because the demand values are computed by assuming the tiles on upper layers have accommodated maximum possible demands. As a result, the prediction map can guide each layer by indicating which tiles require to be assigned more wire segments to avoid overflows on upper layers.

III. The Proposed Flow

In this section, the flow of our concurrent layer assignment framework is given in Section III-A. After that, the four major steps of the framework are separately detailed in Sections III-B–III-E.

A. Concurrent Layer Assignment Framework

The proposed flow of the concurrent layer assignment framework is given in Fig. 6. Inputs are the 2D routing patterns of all nets, and each 2D pattern is decomposed into 1D wire segments according to its turning points. At the beginning, the capacity map and the demand prediction map are initialized. Then, the wire segment assignment process is performed from the lowest available layer to the highest layer, and each iteration is composed of the following four steps:

(i) **Pin Initialization:** The pins of nets may lie on different layers because of the existence of macros or the advanced design of standard cells. Before assigning wire segments, the pins on the current layer are examined and initialized.

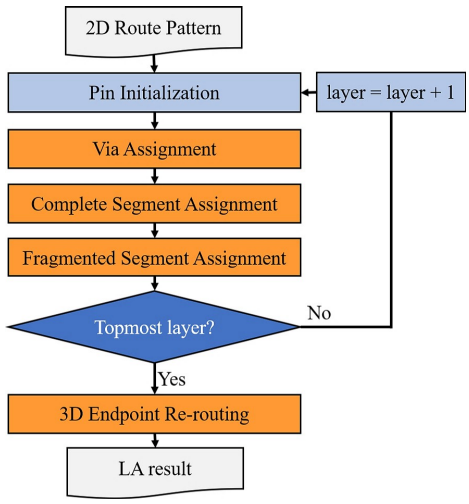

Fig. 6: The proposed framework of concurrent layer assignment.

(ii) **Via Assignment:** To consider via demand for each layer, the vias connecting the current layer and the lower adjacent layer are assigned prior to wire segment assignment.

(iii) **Complete Segment Assignment:** For a target layer, all unassigned candidate wire segments are ordered with three rules, and those with higher priorities will be assigned on the layer without exceeding the capacity of any tile.

(iv) **Fragmented Segment Assignment:** For the remaining routing resource left after complete segment assignment, the unassigned candidate wire segments are fragmented and assigned on the current layer to avoid in advance possible overflows on the topmost layer.

After the layer assignment procedure of the topmost layer is done, 3D endpoint re-routing is required for unassigned wire segments. We will show that due to the concurrent scheme used in our layer assignment approach, and because the proposed layer assignment approach has maximally utilized the routing resources on all layers, the wire segments required to be re-routed are easier to be identified. Furthermore, the rerouting of those unassigned wire segments can be more effective in reducing overflow compared to conventional DP-based layer assignment approaches because of potentially larger solution space. The major steps highlighted in Fig. 6 are detailed in the following subsections.

B. Candidate Segments Creation and Via Assignment

To minimize the number of resulting vias in the iterative layer assignment process from the lowest layer to the highest one, our strategy is to assign wire segments closer to net pins to lower layers and assign those farther away from pins to higher layers. In this way, the shape of a net composed of its wire segments will be like a mountain or a pyramid. To realize the idea, we first determine which wire segments can be assigned to a currently processed layer. A wire segment is defined as a *candidate segment* that can be assigned to the current layer if at least one of its two endpoints has been connected to its pin through the already assigned wire segments and vias on the lower layers. We maintain a segment set S_{seg} during the layer assignment process to store current candidate segments. In addition, to consider via demands for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

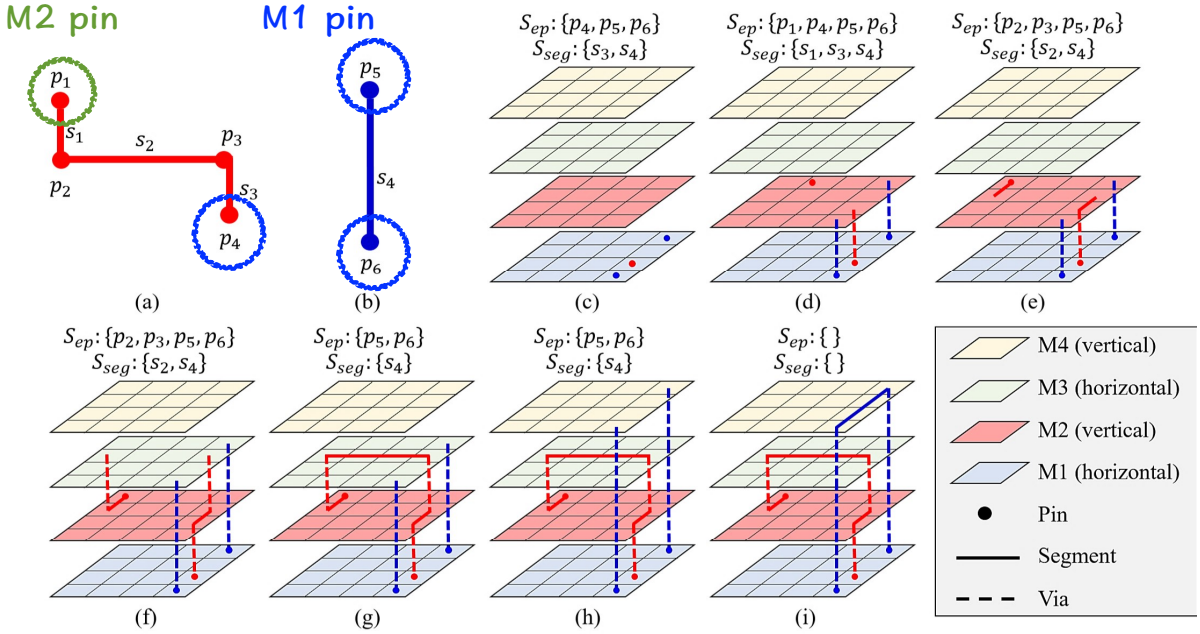

Fig. 7: The process of via assignment and maintaining S_{seg} and S_{ep}. (a) The 2D routing pattern of the red net. (b) The 2D routing pattern of the blue net. (c) Steps (i) and (ii) on M1. (d) Steps (i) and (ii) on M2 and Via Assignment. (e) Steps (iii) and (iv) on M2. (f) Via Assignment and Steps (i) and (ii) on M3. (g) Steps (iii) and (iv) on M3. (h) Via Assignment and Steps (i) and (ii) on M4. (i) Steps (iii) and (iv) on M4.

Layer M_l, we assign the vias connecting Layer M_l and Layer M_{l-1} prior to assigning wire segments on Layer M_l. The via assignment procedure is actually done by deducting the via demands from the tile capacities, and thus wire segments can only be assigned by using the remaining capacities. Because vias are used to connect the endpoints of wire segments, we also maintain an endpoint set S_{ep}. Each endpoint in S_{ep} will result in one via (and thus one unit of demand) on M_l.

For each layer, we use the following three steps to maintain S_{seg} and S_{ep}:

(i) Add all the pins on the current layer into S_{ep}.
(ii) Identify new candidate wire segments connecting to the pins and add them into S_{seg}.
(iii) For each wire segment assigned to the current layer, we remove it from S_{seg} and check the two endpoints for the segment according to the following two scenarios:
- If an endpoint is not in S_{ep}, we add the endpoint to S_{ep}, and add all the segments connecting to the endpoint to S_{seg}.
- If an endpoint is already in S_{ep}, we further check whether there is any other wire segment connecting to the endpoint and has not been assigned to any layer. If the situation is true, we keep the endpoint in S_{ep} as it will continue to cause a via toward the upper layer; otherwise, we remove the endpoint from S_{ep}.

We use the layer assignment instance in Fig. 1 to demonstrate the process of via assignment and maintaining S_{seg} and S_{ep}. The 2D patterns of the two nets are given in Figs. 7(a) and 7(b) with denoted wire segments and endpoints, where p_1 is an M2 pin and p_4, p_5, p_6 are M1 pins.

Starting from M1, as illustrated in Fig. 7(c), the pins p_4, p_5 and p_6 are added to S_{ep} and their connected segments, s_3 and s_4, are added to S_{seg}. Since no wire segment can be assigned to M1, Step 3 is omitted here. Moving to M2, the pin on M2 p_1 is added to S_{ep}, and its connected wire segment S_1 is added to S_{seg}, as shown in Fig. 7(d). Before tackling the wire segments on M2, we assign vias connecting M1 and M2 and deduct the demands in S_{ep} from the tile capacities on M2. Note that in the rightmost column, due to capacity deduction by vias, s_4 cannot be directly assigned to M2. As a result, only s_1 and s_3 are assigned to M2, as shown in Fig. 7(e). After assigning s_1 and s_3, the two segments are removed from S_{seg}, and p_1 and p_4 are removed from S_{ep}. In addition, p_2 and p_3 are added to S_{ep}, and s_2 also becomes a candidate wire segment. Moving to M3, vias are first assigned, as illustrated in Fig. 7(f), and s_2 is assigned on M3, as shown in Fig. 7(g). Since all the connected segments of the two endpoints of s_2 have been assigned on layers, p_2 and p_3 are removed from S_{ep}. Finally, the two vias at p_5 and p_6 in S_{ep} and the remaining segment s_4 in S_{seg} are assigned to M4, as respectively shown in Figs. 7(h) and 7(i).

Note that in this example, we only demonstrate how to manipulate S_{seg} and S_{ep} when initializing a layer and assigning a wire segment to a layer. The details of determining which candidate segments are assigned to the current layer are detailed in Sections III-C and III-D.

C. Complete Segment Assignment

There are two ways to assign a segment on a layer. The first one is to assign a complete candidate segment in S_{seg} to the current layer. Another way is to fragment a candidate segment and only assign a part of the segment. To minimize the number of vias while maximizing the utilization of routing resource, we first assign *complete segments* as many as possible and then fill up the remaining capacities with *fragmented segments*. The candidate segments in S_{seg} of the current layer are sorted according to three criteria and are sequentially assigned. Once no complete segment can be further assigned, we check whether any new candidate segment is generated through the

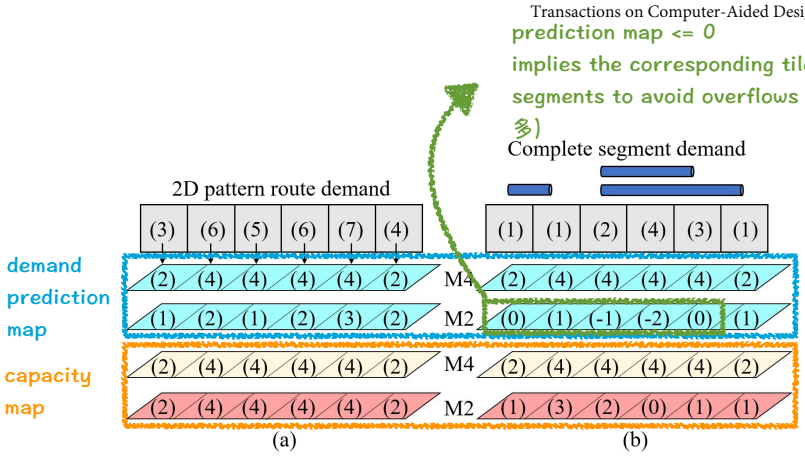

Fig. 8: Prediction map update for fragmented segments. (a) A column of 2D demands and the initialized demand prediction map and capacity map. (b) The updated demand prediction map and capacity map after complete segment assignment.

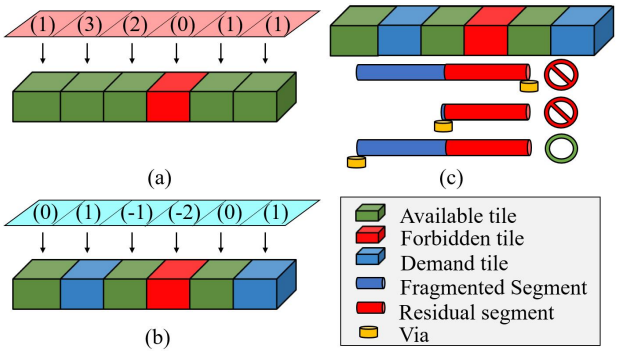

Fig. 9: Fragmented segment ordering and assign into the left sub-column. (a) Tile statuses according to remaining capacities. (b) Tile statuses according to predicted demand values. (c) The third segment can be fragmented into blue segment and assigned.

new endpoints added to S_{ep}. If the situation is true, the generated segments are added to S_{seg}, and the process is iteratively performed until no complete segment is left or no complete segment can be assigned due to insufficient capacity.

The complete segments are sorted according to the following three criteria:

(i) **Residual parts of fragmented segments:** If a candidate segment is the residual part of a fragmented segment, it has the highest priority to be assigned. It is because the incompleteness of a fragmented segment prevents other segments of the same net from becoming candidate segments.

(ii) **The degree of net completeness:** If most of the segments of a net have been assigned on layers, the unassigned segments of the net should have higher priorities to be assigned. It is because if all the segments of a net have been assigned on layers, the net will no longer cost any via demand and thus the number of vias can be minimized. The degree of completeness of a net is calculated as follows:

$$Completeness = N_{assigned}/N_{total}, \tag{3}$$

where $N_{assigned}$ is the number of segments that have been assigned, and N_{total} is the total number of segments of the net in the original 2D global routing result.

(iii) **Segment length:** If two segments have the same degree of net completeness, we break the tie according to their segment lengths and assign the shorter segment prior to the longer one. It is because for the same capacity, the number of shorter segments can be assigned is usually more than that of longer segments, which results in more nets whose segments can be completely assigned on lower layers. Thus, assigning shorter segments prior to longer segments can result in fewer number of vias.

D. Fragmented Segment Assignment

After assigning complete segments as many as possible, the remaining scattered routing resource can still be filled with fragmented candidate segments. First, the available tiles with non-negative capacities are identified and the prediction map is updated to guide the fragmented segment assignment process. After that, the fragmented segments are sorted and assigned in order until no segment is left or running out of capacities. Finally, the resulting prediction map of the current layer is used to update the prediction map of the upper layer in the same direction. The three steps are detailed in the following.

1) Prediction map update for fragmented segments: There may still be many scattered capacities after complete segment assignments, while how to effectively use these remaining resources is crucial for via and overflow minimization. As introduced in Section II-B, the values in a demand prediction map on the current layer indicate the minimum demand each tile needs to be assigned to avoid overflow occurring on the topmost layer. Thus, we update the prediction map after complete segment assignment to guide the following fragmented segment assignment process. An example column of tiles is illustrated in Fig. 8, where the 2D demands, the 3D predicted demands, and the 3D capacities of the tiles before complete segment assignment are given in Fig. 8(a). Suppose three complete segments shown in Fig. 8(b) have been assigned to M2, which consume the demands by $\{1, 1, 2, 4, 3, 1\}$ from the leftmost to the rightmost tiles. The consumed demands reduce the capacities of tiles from $\{2, 4, 4, 4, 4, 2\}$ to $\{1, 3, 2, 0, 1, 1\}$, and the values in the prediction map are also reduced from $\{1, 2, 1, 2, 3, 2\}$ to $\{0, 1, -1, -2, 0, 1\}$. The zero and negative predicted demands after the update indicate that the corresponding tiles have been assigned enough segments to avoid overflows on the topmost layer. In contrast, positive predicted demands imply that more segments should be assigned to the corresponding tiles, and thus fragmented segment assignment is adopted in the following.

2) Fragmented segment ordering and assignment: Focusing on the capacities and demand prediction values on M2 in the example in Fig. 8(b), we define different statuses of each tile. As illustrated in Fig. 9(a), the tiles with remaining capacities (available tiles) are colored in green, and the tile with no capacity left (forbidden tile) is colored in red. In addition, the tiles with positive demand prediction values (demand tiles) are explicitly colored in blue, as illustrated in Fig. 9(b). According to the three tile statuses, a fragmented segment assigned to the current layer cannot overlap red tiles. Besides, it is more desirable to assign fragmented segments overlapping with blue tiles to avoid overflows on the highest layer. Red tiles split a column of tiles into sub-columns, and we focus on a single sub-column at a time. A candidate segment can be fragmented and assigned for a sub-column if the following two conditions

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

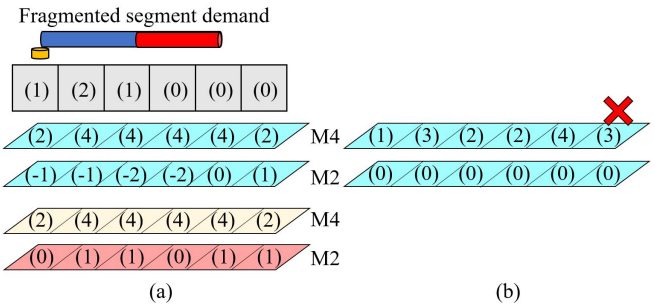

Fig. 10: Prediction map update for the upper layer. (a) The demand of a fragmented and assigned wire segment. (b) The demand prediction values on M4 are updated with those on M2.

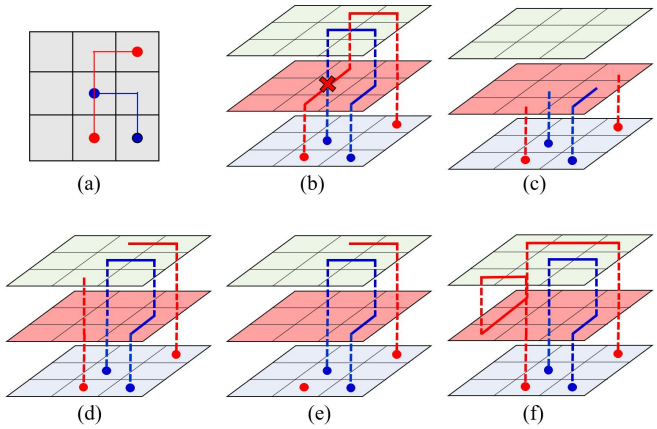

Fig. 11: 3D Re-routing (a) Feasible 2D global routing paths. (b) One overflow must occur no matter which layer assignment approach is adopted. (c) The assignment result on M2 with the concurrent method. (d) The assignment result on M3 with the concurrent method. (e) Redundant via removal. (f) The 3D re-routing result without any overflow.

are satisfied: (1) its connected via to the lower layer lies in the sub-column, and (2) its fragmented sub-part (blue part) has to overlap the sub-column with more than one tile. An example is given in Fig. 9(c), where the red tile splits the column into two sub-columns, and the left/right one consists of three/two tiles. For the left sub-column, the first and second segments cannot be assigned because the connected via of the first segment does not lie in the left sub-column, and the second segment only overlaps the left sub-column with one tile. The third segment satisfies the two conditions and thus can be fragmented, whose blue sub-part can be assigned on the left sub-column, and the red sub-part becomes a candidate segment in S_{seg}.

We determine the ordering of candidate segments to be fragmented according to the following gain:

$$Gain = \alpha_1 \times DT + \beta_1 \times AT \tag{4}$$

, where DT and AT are respectively the number of demand (blue) tiles and the number of available (green) tiles a segment overlaps, and α_1 and β_1 are user-defined weights. The DT and AT of the third segment in Fig. 9(c) are respectively one and three. The candidate segment with the highest gain has the first priority to be fragmented and assigned. In our experiments, α_2 and β_2 are respectively set to 20 and 1.

3) Prediction map update for the upper layer: After no more fragmented segment can be assigned, the prediction map of the current layer is used to update that of the upper layer in the same direction. An example is shown in Fig. 10, where only one fragmented segment is assigned to M2. Due to the demand consumed by the segment, the demand prediction values on M2 is reduced from $\{0, 1, -1, -2, 0, 1\}$ in Fig. 9(b) to $\{-1, -1, -2, -2, 0, 1\}$ in Fig. 10(a). The demand prediction values on M2 are then added to those on M4, and the updated demand prediction values on M4 become $\{1, 3, 2, 2, 4, 3\}$. Note that if a prediction value increases after the update, it implies that overflow must occur in the tile on the highest layer, as the rightmost tile in Fig. 10(b)

E. 3D Endpoint Re-routing

After assigning wire segments for the topmost layer, some segments may still be left unassigned, which are due to insufficient routing resource in congested regions. A congestion situation may be unexpected during 2D global routing because of an inaccurate 2D capacity and demand model. Fig. 11(a) illustrates an example, where the edge capacity is one and the 2D global routing patterns of the two nets do not cause any overflow. However, as shown in Fig. 11(b), no matter which layer assignment algorithm is adopted, an overflow must occur. This is because the actual via demand cannot be accurately predicted until segments are assigned to specific layers. To effectively reduce these overflowing segments, we adopt a 3D endpoint re-routing process, which is composed of the following four steps:

(i) **Redundant Via and partially fragmented segment Removal:** The first step is to identify the unassigned wire segments and remove their redundant vias and partially fragmented segments. For the layer assignment instance in Fig. 11(a), the layer assignment results on Layer 2 and Layer 3 are respectively shown in Figs. 11(c) and (d). Since a red wire segment fails to be assigned, its connected via is removed, as shown in Fig. 11(e). Removing redundant vias of unassigned segments can enlarge the rerouting flexibility and thus may better improve wire overflow.

(ii) **3D Multi-Endpoint Decomposition:** After removing redundant vias, there may be more than two segment endpoints of a net needing to be connected by rerouting. To first decompose such a multi-endpoint connection into two-enpoints nets, we find a 3D Rectilinear Minimum Spanning Tree (RMST) among these segment endpoints, where the edge cost of two endpoints in the spanning graph is set as their 3D Manhattan distance.

(iii) **3D Net Ordering:** To determine the ordering to route these two-endpoint nets, we compute the cost of Net i as follows:

$$Cost_i = \alpha_2 \times WL + \beta_2 \times Con, \tag{5}$$

where α_2 and β_2 are user-defined parameters, WL is the same 3D Manhattan distance as that in the spanning graph, and Con indicates the congestion in the 3D bounding box of a two-endpoint net, which is equal to the total remaining capacity divided by the total number of global tiles in the bounding box. A 2-endpoint net with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

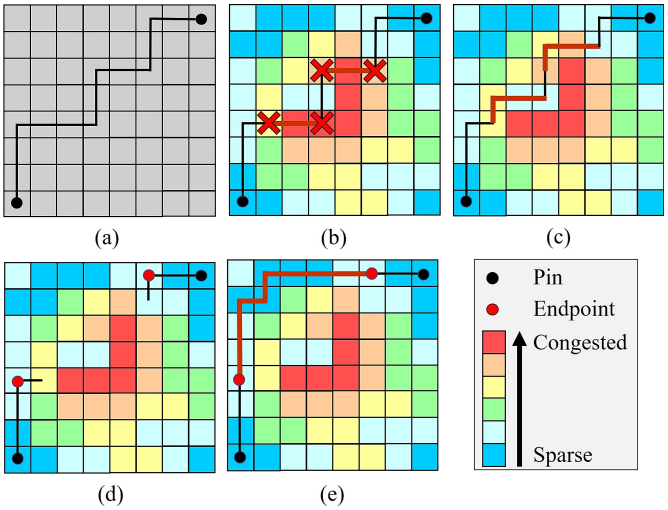

Fig. 12: An example explaining the difference of re-routing between DP-based approaches and our method. (a) A 2D global routing pattern. (b) The segments with overflows are individually ripped-up and rerouted in DP-based methods. (c) The re-rerouting segments still cause few overflows, because the nearby regions are still congested. (d) With our concurrent layer assignment, there are some segments left un-assigned and thus un-routed. (e) The endpoint re-routing of the un-assigned segments can find a less-congested path.

smaller cost has higher priority to be re-routed due to its limited routing resource and routing flexibility. In our experiments, α_2 and β_2 are respectively set to 1 and 100.

(iv) **3D Endpoint Rerouting:** Determining 3D endpoint net ordering, we route each two-endpoint net with the A^* search algorithm. In the first iteration, each net is routed inside its 3D bounding box, and overflow is not allowed. If any net is unsuccessfully routed, the bounding box will be slightly expanded and one tolerated overflow will be incrementally added for each net. The iterative rerouting procedure will be terminated once all nets are successfully routed or the number of iterations reaches the specified limit. If the later situation happens, the overflow toleration will be unlimited. The rerouting result of the red net in the example is given in Fig. 11(f).

It is worth to mention that the 3D endpoint re-routing process proposed here is greatly different from the conventional rip-up and rerouting process used after a sequential layer assignment approach. After our concurrent layer assignment step, there is no overflow on every metal layer; instead, some wire segments are not assigned onto any layer because their assignments will cause overflow on the topmost layer. As a result, the process introduced in this section only reroutes the un-assigned (un-routed) segments without ripping-up any segments that have been assigned, which is one of the major advantages of the proposed concurrent layer assignment scheme.

The other advantage of the proposed method is that it has an extraordinary ability to guide the 3D endpoint re-routing process and thus can effectively reduce the total overflow. Fig. 12 explains the reason. Fig. 12(a) shows a net with six wire segments, where two of them pass through congested region with inevitable overflows, as illustrated in Fig. 12(b). We need to emphasize that the overflows after layer assignment may be unexpected during 2D global routing due to inaccurate via demand models used in 2D global routers. As we have mentioned in Section I, a conventional DP-based layer assignment algorithm sequentially finds the layer assignment solution for each net, which causes another ordering problem in the refinement stage, and each wire segment suffering from overflow can only be individually ripped up and rerouted in congested regions, as illustrated in Fig. 12(c). In contrast, since our concurrent layer assignment scheme assigns the segments of a net from the bottommost layer to the topmost layer, the un-assigned segments of a net is continuous, as shown in Fig. 12(d). Thus, the rerouting of the unassigned segments may find a less-congested path due to the larger solution space, as illustrated in Fig. 12(e).

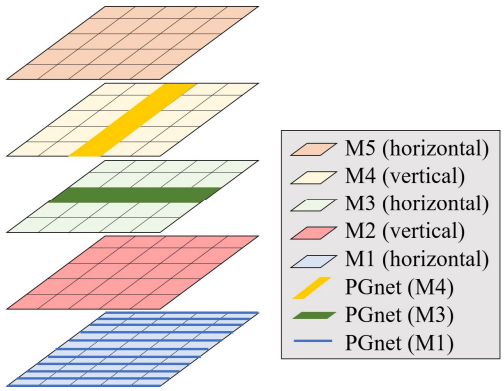

Fig. 13: PG nets on different layers.

It may be observed in Fig. 12 that if a DP-based approach rips up the whole 2-pin net and reroutes it, the same refinement solution in Fig. 12(e) can be derived. However, ripping up the segments without overflow and rerouting them elsewhere not only requires additional routing resources but also wastes the original resources found during the DP optimization process. The wasted resources are usually disjointed and can hardly be used by other re-routed nets, which may lead to even more overflows. As a result, ripping up whole nets is usually not considered in the DP-based approaches. In contrast, the segments re-routed after our concurrent layer assignment process are originally unassigned. Thus, no step of ripping-up is required and no resource will be wasted. In summary, the obvious difference between our 3D endpoint rerouting and the conventional rip-up and re-routing is that the conventional method focuses on perturbing the segments *inside* a congestion region with overflow, while ours does not generate any overflow before the endpoint re-routing process and only re-routes un-assigned segments in sparse area *outside* congestion regions.

IV. Obstacle-aware Strategy

In the current physical design process in large-scale designs, obstacles present before routing for signal nets, which could result in great overflows if they are not considered during layer assignment. Huge obstacles such as power and ground (PG) nets on upper layers especially occupy large routing resources in local regions. Fig. 14(a) shows the layout of ispd19_test1 in the 2019 ISPD Initial Detailed Routing Contest before signal routing. We highlight the local region in the yellow box and separately show the PG nets on M1, M2, M5, M7, and M8 in Figs. 14(b)–(f), where gray grids indicate global tiles. It

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

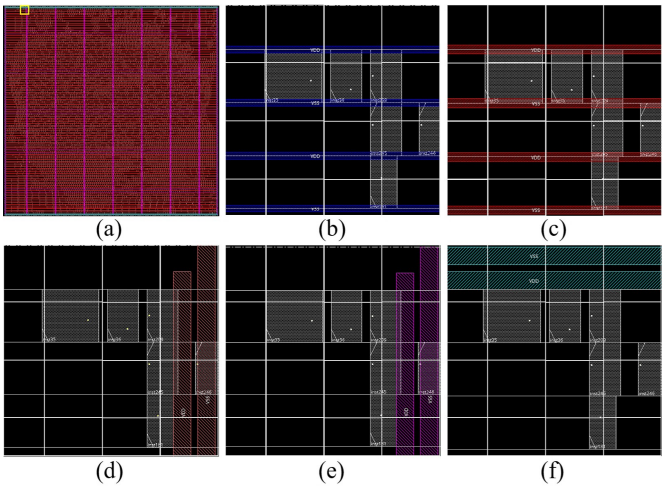

Fig. 14: PG nets in the benchmark ispd19_test1. (a) The layout of ispd19_test1. (b) PG nets on M1. (c) PG nets on M2. (d) PG nets on M5. (e) PG nets on M7. (f) PG nets on M8.

can be observed that PG nets on the lower layers are thin and occupy fewer routing resources, while PG nets on the upper layers are thick and may block most of the routing resource in a global tile.

In the previous sections, we have proposed a robust and concurrent layer assignment scheme (the basic COALA framework). Despite that the proposed demand prediction map can predict whether the overflow will occur in a tile row on a current layer, it cannot effectively avoid overflows caused by obstacles on the upper layers. To solve the problem, we further propose an obstacle-aware strategy compatible with the basic COALA framework. The following subsections respectively introduce the adopted 3D obstacle map considering different kinds of obstacles and explain how to consider obstacles in our framework.

A. 3D Obstacle Map

Fig. 13 illustrates an example of routing resource reduction due to obstacles, where M1 has thin PG nets, M3 and M4 have thick PG nets merely occupying a row or a column of global tiles, and M2 and M5 are free routing layers without pre-routed PG nets. To address different obstacle occupation statuses in global tiles, we additionally propose a 3D obstacle map, which records three different types of obstacles as follows:

(i) **Obstacle:** Each obstacle of this type occupies most of routing resource in a global tile, such as macro obstacles, macro PG nets, and thick PG nets.
(ii) **Barrier:** Each obstacle of this type occupies about half of the routing resource in a global tile, such as cell obstacles.
(iii) **PG:** Each obstacle of this type occupies a small part of the routing resource in a global tile, such as cell PG nets and thin PG nets on lower layers.

The 3D obstacle map is initialized as a 3D vector, and each entry records a score $S(n,i,j)$ for the global tile $T(n,i,j)$ on layer n and with x- and y-coordinate (i,j), which indicates the degree of obstacle occupation in the tile. The score is computed as follows:

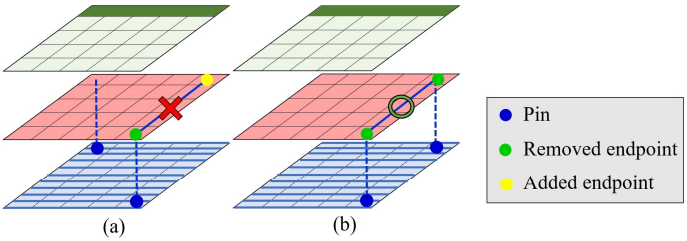

Fig. 15: Inevitable overflow segment forbidden. (a) An obstacle is above the added endpoint, and thus we forbid assigning the segment to the layer. (b) The segment is assignable because both the two endpoints of the segment are removed endpoints.

$$S(n,i,j) = w_{obs} \times Obs(n,i,j) + w_{bar} \times Bar(n,i,j) + w_{pg} \times PG(n,i,j), \quad (6)$$

where $Obs(n,i,j)$, $Bar(n,i,j)$ and $PG(n,i,j)$ are binary variables indicating whether any obstacle, barrier, or/and PG net exist in $T(n,i,j)$. w_{obs}, w_{bar}, and w_{pg} are user-defined parameters for the three types of obstacles, which are separately 4, 2, and 1 in our implementation. As a result, each $S(n,i,j)$ in the obstacle map is ranged from 0 to 7, which encodes whether and what types of obstacles exist in $T(n,i,j)$, and a larger score represents that more obstacles are in a tile. In our experiments, the capacity of each tile is deducted if obstacles exist, which is done by multiplying a value between 0 to 1. For $S(n,i,j)$ that is larger than or equal to 4, the multiplier is 0 and thus the remained capacity is zero. For $S(n,i,j)$ that is equivalent to 3, 2, 1 and 0, the multipliers are separately 0.1 , 0.2, 0.9 and 1.

B. Obstacle-aware Techniques

With the assist of the 3D obstacle map, our goal is to avoid segments as well as their connected vias inevitably bumping into obstacles by well-assigning segments on free routing layers. For better explanation, we first define some terminologies. Every candidate segment has two endpoints, and an endpoint connected from the lower layer is referred to as a *removed endpoint*, since it will be removed from S_{ep} if its connected segments are all assigned on layers. Similarly, an endpoint connected to the upper layer is referred to as an *added endpoint*, since it will be added to S_{ep} if the candidate segment is assigned to the current layer. Two strategies are adopted, which are detailed in the following:

(i) **Escape from obstacle regions:** This method gives segments that are more relevant to the 3D obstacle map higher priorities to be assigned on lower layers. More precisely, it is more desirable to assign a segment on lower layers whose two endpoints lie in two global tiles with many obstacles, and thus the segment could escape from obstacle regions as early as possible. The relevance score of a segment is calculated as follows:

$$Score_{rel} = \sum_{n=l_{cur}+1}^{l_{top}} w(l_{cur},n) \times (S(n,i,j) + S(n,k,l)), \quad (7)$$

where the x- and y-coordinates of the two endpoints of the segment are (i,j) and (k,l), l_{cur} is the index of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

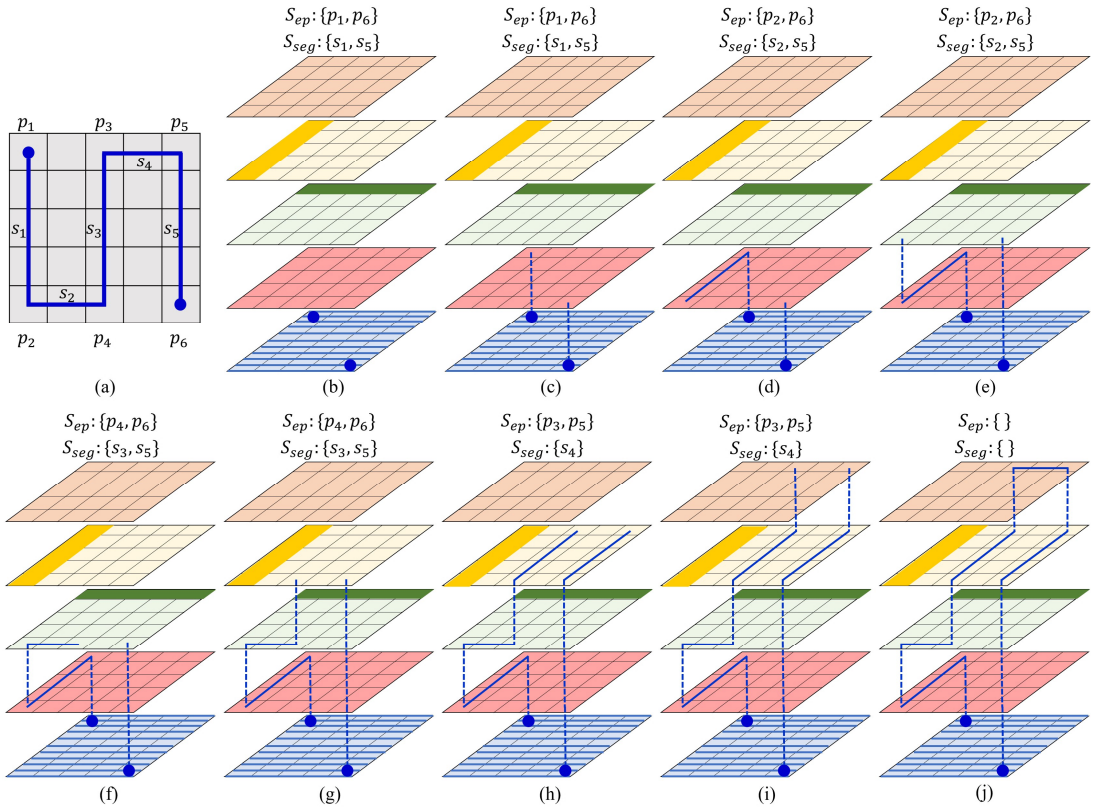

Fig. 16: The process of COALA with obstacle awareness. (a) The 2D routing pattern of the blue net. (b) Via assignment on M1. (c) Via assignment on M2. (d) Candidate segment assignment on M2. (e) Via assignment on M3. (f) Candidate segment assignment on M3. (g) Via assignment on M4. (h) Candidate segment assignment on M4. (i) Via assignment on M5. (j) Candidate segment assignment on M5.

the current layer, and l_{top} is the index of the topmost layer. $w_{(l_{cur},n)}$ represents the weight determined by the distance between the current layer and Layer n, which is calculated in our implementation as follows:

$$w(l_{cur}, n) = l_{top} - (l_{cur} - n). \tag{8}$$

With this equation, the closer Layer n is from the current layer, the greater the weight is set. The relevance score represents how critical and eager a segment is to be assigned on lower layers to avoid overflow on upper layers due to the existence of obstacles.

(ii) **Forbid inevitable overflow assignment:** This method forbids segment assignment that must cause overflow on upper layers. For a candidate segment assignable on Layer l, if a removed endpoint of the candidate segment is below a global tile fully occupied by obstacles on Layers $l+1$ or $l+2$, the segment has the highest priority to be assigned on Layer l. It is because if the candidate segment is not assigned on Layer l, it or/and its via must cause overflow on Layer $l+1$ or $l+2$. In contrast, if an added endpoint of a candidate segment is below a global tile fully occupied by obstacles on Layers $l+1$ or $l+2$, the segment is not allowed to be assigned on Layer l. Fig. 15(a) shows an example where the segment is not allowed to be assigned on M2 to avoid inevitable overflow. Fig. 15(b) shows another similar example while the segment is fine to be assigned. It is because the two endpoints of the segment in Fig. 15(b) are both removed endpoints, and thus no via is required to connect the segment to the upper layer.

Note that in our implementation, the two strategies are considered prior to those in Sections III-C and III-D. That is, "forbid inevitable overflow assignment" is first performed, and "escape from obstacle regions" is then considered, where all segments have non-zero relevance scores have higher priorities than those with zero scores. The segments with the same score are then ordered by using the criteria in Sections III-C and III-D.

We use the layer assignment instance in Fig. 16 to demonstrate the two obstacle-aware strategies. The 2D pattern is given in Fig. 16(a). Different from the example in Fig. 7 there are PG nets on M1 and M3. Following Section III-B, two M1 pins are not for layer assignment as shown in Figs. 16(b) and (c), and $p1$, $p6$, s_1, and s_5 are added to S_{ep} and S_{seg}. According to the two obstacle-aware strategies, the candidate segment $s1$ has very high relevance score and thus is assigned on M2, and $s5$ is not allowed to be assigned on M2 because its assignment will result in an inevitable overflow due to the PG net on M3, as shown in Figs. 16(d) and (e). The following layer assignment process is illustrated in Figs. 16(f)–(j), where it can be observed that $s5$ is finally assigned on M5 to achieve an overflow-free assignment.

V. Experimental Results

We implemented our algorithm flow with the C++ programming language and all experiments were performed on a 3.5

TABLE I: The layer assignment results of the original ISPD-2018 benchmarks, and the results after applying the DP-based method using Technique (i), and the DP-based method using Techniques (i)+(ii).

Case name	ISPD-2018 [25]			DP-based with Technique (i) [7]				DP-based with Technique (i)+(ii) [7]+[3]			
	OF MAX	OF NUM	WL	OF MAX	OF NUM	WL	CPU	OF MAX	OF NUM	WL	CPU
ispd18_test1	0	0	63869	2	2	70008	1	1	2	66237	0.8
ispd18_test2	0	0	930846	3	42	1011846	21	3	31	959957	19
ispd18_test3	25	3698	1011828	14	2423	1067594	27	14	2381	1010064	25
ispd18_test4	17	20019	2737564	22	14320	3128047	128	22	14153	2927401	113
ispd18_test5	17	67230	3055257	30	59186	3460974	144	30	56040	3326776	132
ispd18_test6	10	79493	4224656	11	82450	4734593	155	11	77021	4582360	137
ispd18_test7	16	173110	7360506	33	169749	8389528	522	33	161233	8114366	482
ispd18_test8	16	192990	7405141	36	176781	8508146	513	36	167529	8208920	470
ispd18_test9	14	169887	6631012	14	155238	7678465	225	14	146349	7384515	197
ispd18_test10	24	203206	7717030	15	167316	8796543	293	15	157132	8483040	260
Comp	1.727	1.363	0.932	2.080	1.383	1.045	0.816	1.980	1.295	0.999	0.729

TABLE II: The layer assignment results of the ISPD-2018 benchmarks for the DP-based method using Techniques (i)+(ii)+(iii), COALA without obstacle awareness, and COALA with obstacle awareness.

Case name	DP-based with Technique (i)+(ii)+(iii) [7]+[3]+[13]				COALA w/o Obstacle Awareness				COALA w/ Obstacle Awareness				
	OF MAX	OF NUM	WL	CPU	OF MAX	OF NUM	WL	CPU	OF MAX	OF NUM	WL	CPU	RR%
ispd18_test1	1	2	66237	1.06	0	0	65656	0.7	0	0	65612	0.7	0.00%
ispd18_test2	2	24	959985	24	1	9	949234	9	1	1	948313	9	0.09%
ispd18_test3	3	1961	1010361	31	4	1640	1001733	12	2	1689	1002218	12	0.79%
ispd18_test4	9	4276	2937102	147	5	2781	2892020	58	3	2981	2891468	56	2.88%
ispd18_test5	16	53691	3328564	188	11	56577	3176831	72	10	44233	3243066	73	3.88%
ispd18_test6	11	76990	4582450	183	12	84295	4361312	72	11	64362	4470618	74	1.90%
ispd18_test7	12	155223	8117179	727	15	165646	7670849	449	11	135240	7864882	444	6.45%
ispd18_test8	14	161057	8212054	713	15	171161	7753383	354	13	137845	7955183	342	6.82%
ispd18_test9	14	146296	7384687	302	14	159254	6952927	157	12	123756	7151603	162	2.76%
ispd18_test10	15	155367	8487496	366	16	167300	8026958	166	14	131744	8232865	169	3.69%
Comp	1.000	1.000	1.000	1.000	0.855	0.830	0.963	0.471	0.676	0.668	0.978	0.472	2.93%

TABLE III: The layer assignment results of the original ISPD-2019 benchmarks, and the results after applying the DP-based method using Technique (i), and the DP-based method using Techniques (i)+(ii).

Case name	ISPD-2019 [26]			DP-based with Technique (i) [7]				DP-based with Technique (i)+(ii) [7]+[3]			
	OF MAX	OF NUM	WL	OF MAX	OF NUM	WL	CPU	OF MAX	OF NUM	WL	CPU
ispd19_test1	2	2	83925	5	20	91907	1.6	3	7	85042	1.5
ispd19_test2	20	6530	2827886	12	12696	3043352	145	12	7779	2899721	131
ispd19_test3	24	1923	154673	19	3247	170767	2.5	19	2827	161825	2.2
ispd19_test4	30	23412	3257251	54	16758	3106071	209	55	16464	3076189	202
ispd19_test5	29	9971	542324	48	6201	515973	16	48	6046	505719	16
ispd19_test6	27	22909	7378089	46	26138	7624866	709	46	23403	7207402	661
ispd19_test7	19	31494	14469077	32	42995	14327487	1993	32	19239	13532956	1916
ispd19_test8	19	188249	21894715	34	404654	23310392	3072	34	365119	22394715	2923
ispd19_test9	22	296491	34555150	32	684898	36894413	5595	32	620699	35369302	5352
ispd19_test10	26	770140	34865163	43	511512	36136292	5661	43	426924	34562579	5439
Comp	2.034	1.316	1.007	2.923	1.694	1.043	0.901	2.826	1.210	0.997	0.846

TABLE IV: The layer assignment results of the ISPD-2019 benchmarks for the DP-based method using Techniques (i)+(ii)+(iii), COALA without obstacle awareness, and COALA with obstacle awareness.

Case name	DP-based with Technique (i)+(ii)+(iii) [7]+[3]+[13]				COALA w/o Obstacle Awareness				COALA w/ Obstacle Awareness				
	OF MAX	OF NUM	WL	CPU	OF MAX	OF NUM	WL	CPU	OF MAX	OF NUM	WL	CPU	RR%
ispd19_test1	2	7	85046	1.4	3	12	84169	1	0	0	84107	1	0.00%
ispd19_test2	5	6665	2900417	134	7	8708	2874799	53	5	8317	2873135	53	1.47%
ispd19_test3	15	2777	161979	2.6	14	2478	157736	2.3	13	2361	158496	2.5	1.62%
ispd19_test4	30	10568	3092370	467	29	8545	3017108	469	30	8322	3011796	267	31.54%
ispd19_test5	30	3524	514560	64	31	3553	516102	92	32	3682	517292	47	26.09%
ispd19_test6	9	16257	7212749	642	3	11723	7116459	309	4	10535	7117262	316	2.14%
ispd19_test7	4	16485	13534167	1792	5	33584	13420846	775	5	29530	13406000	759	3.89%
ispd19_test8	13	361351	22396461	2797	15	398320	21445590	1567	11	344400	22017607	1550	3.63%
ispd19_test9	14	616500	35369305	5235	16	677910	33688685	3099	12	595874	3415739	3074	3.12%
ispd19_test10	26	417499	34618442	7185	14	414619	33338434	3839	16	303183	33991202	4203	3.47%
Comp	1.000	1.000	1.000	1.000	1.025	1.168	0.978	0.701	0.794	0.901	0.986	0.601	7.70%

GHz Linux machine with 64 GB memory. We use the benchmarks used in 2018 and 2019 ISPD Initial Detailed Routing Contest (denoted as ISPD-2018 and ISPD-2019 respectively) for experiments [25], [26]. The 3D global routing guides provided in the benchmarks are projected and transformed into 2D global routing paths, which serve as the input of each layer assignment approach. More importantly, the benchmarks in the ISPD-2019 contain fixed PG rails that do not exist in the ISPD-2018 benchmarks. The following experimental results are divided into two parts; Section V-A gives the global routing results for different layer assignment approaches in terms of overflow, wirelength, and runtime, and Section V-B gives the detailed routing results with metal utilization, via utilization, and DRVs.

A. Validation with Global Routing Results

We compare our concurrent layer assignment with an implemented DP-based layer assignment approach, which is composed of three techniques similarly adopted in most of the state-of-the-art layer assignment works. The three techniques are as follows:

(i) **DP-based layer assignment considering via demands:** The compared DP-based approach is implemented based on [8], which minimizes the total overflow by using the well-known congestion-constrained layer assignment (COLA) algorithm proposed in [7] and additionally considers via demands in their capacity model.

(ii) **Net decomposition:** Since larger nets usually have lower priorities to be assigned, [3] proposes to decompose a large net into sub-nets, which enhances the flexibility and solution space of net ordering in the sequential layer assignment approaches.

(iii) **3D Rip-up and Rerouting:** Most of the solution refinement techniques proposed in existing studies rip up wire segments suffering from overflows and reassign them without changing global routing paths. For fair comparison, we apply the 3D rip-up and rerouting process proposed in [13]. Note that comparing with our 3D endpoint re-routing technique, the conventional rip-up and rerouting process spends lots of time on iteratively ripping up different sets of segments and rerouting them, while our approach can focus on re-routing those unassigned segments. The obvious runtime gap can be shown in the following results.

The layer assignment results are shown in Tables I– IV, where "OF MAX" represents the maximum overflow value in a tile, "OF NUM" represents the total number of tiles with overflows, "WL" represents the total wirelength whose unit is the number of tiles, "CPU" represents the running time in seconds, and "%RR" represents the ratio of the number of unassigned segments to the number of toal segments before 3D endpoint re-routing.

Table I gives the layer assignment results of the original ISPD-2018 benchmarks, the DP-based method using Technique (i), and the DP-based method using Techniques (i)+(ii). Table II gives the results of the DP-based method using Techniques (i)+(ii)+(iii), the COALA framework without the obstacle-aware strategy and the COALA framework with the obstacle-aware strategy. Similarly, Table III and Table IV respectively give the results of the six different methods for the ISPD-2019 benchmarks.

By observing Table I and Table II, the three methods in Table I have similar overflow results, showing that sequential layer assignment approaches suffers from limited solution quality. By adopting the conventional 3D rip-up and rerouting technique, the DP-based layer assignment results can be greatly improved, as the results shown in the column "DP-based with Technique (i)+(ii)+(iii)" in Table II. Comparing the best results of the DP-based layer assignment approach (with 3D rip-up and re-routing), our COALA framework without obstacle awareness can respectively reduce the maximum overflow and overflow tiles by 14% and 16%, and COALA also reduces the wirelength by 3% with only 47% of runtime usage, as shown in Table II. Furthermore, by adopting the obstacle-aware strategy, the reductions in the maximum overflow and overflow tiles can achieve 32% and 33%, with 2% wirelength reduction and similar runtimes. The results show that compared to the sequential layer assignment approaches, the concurrent layout assignment scheme can greatly reduce overflows with much less runtimes, and the obstacle-aware strategy can further improve the overall performance.

In the ISPD-2019 benchmarks where PG rails present, as shown in Table III and Table IV, the COALA framework without obstacle consideration shows 18% larger maximum overflow and 24% more overflow tiles compared to the best results of the DP-based approach. Here the DP-based approach outperforms COALA because it assigns the wire segments of each net by minimizing overflows, and thus the segments will not overlap with PG rails if an overlap-free solution exists. In contrast, the basic COALA framework without obstacle awareness can easily result in wire segments inevitably overlapping with PG rails, which leads to worse results even if 3D endpoint re-routing is adopted. By considering obstacles especially PG rails in the ISPD-2019 benchmarks and comparing to the best results of the DP-based approach, COALA averagely reduces the maximum overflow by 20%, the number of overflow tiles by 9%, wirelength by 1%, and runtimes by 39%. This huge improvement in overflows shows the effectiveness and necessity of adopting the proposed obstacle-aware strategy in COALA. Note that the the average proportions of un-assigned segments requiring 3D endpoint re-routing are respectively about 3% and 8% in ISPD-2018 and ISPD-2019 benchmarks.

B. Validation with Detailed Routing Results

To further verify the effectiveness of our method, we compare the detailed routing results between our concurrent layer assignment and the DP-based layer assignment approach. The detailed router we used in this paper is TritonRoute from "The OpenROAD Project" [27]. The numbers of DRVs are reported by Innovus Implementation System 20.10.000 [28]. We import the global guides generated by the two methods into TritonRoute. Then, the detailed routing results are generated in the DEF file from TritonRoute, whose DRVs can be calculated by Innovus.

The detailed routing results of the two benchmarks are respectively shown in Tables V and VI, where "Itr" represents the numbers of iterations of detailed routing, "IDV" represents the total number of initial DRVs after track assignment, "DV" represents the total number of DRVs after detailed routing, "MS" the number of metal short violations, "CS" the nubmer of metal corner spacing violations, "EL" the end-of-line spacing violations, "PL" the number of parallel run length spacing violations, "AS" the number of adjacent cut spacing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2022.3178353, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE V: The detailed routing results of the ISPD-2018 benchmarks for the DP-based method using Techniques (i)+(ii)+(iii), and COALA with obstacle awareness.

Case name	DP-based with Technique (i)+(ii)+(iii) [7]+[3]+[13]										COALA w/ Obstacle Awareness									
	Itr	IDV	DV	MS	CS	EL	PL	AS	MA	CPU	Itr	IDV	DV	MS	CS	EL	PL	AS	MA	CPU
ispd18_test1	4	3653	0	0	0	0	0	0	0	142.0	7	3450	0	0	0	0	0	0	0	161.2
ispd18_test2	25	45014	0	0	0	0	0	0	0	1859.2	5	42087	0	0	0	0	0	0	0	2344.3
ispd18_test3	19	50568	0	0	0	0	0	0	0	2039.49	18	49032	1	0	0	0	0	0	1	3365.7
ispd18_test4	20	175995	0	0	0	0	0	0	0	2008.0	12	165852	0	0	0	0	0	0	0	2304.4
ispd18_test5	10	323408	0	0	0	0	0	0	0	2398.0	11	299685	0	0	0	0	0	0	0	2590.7
ispd18_test6	5	500508	0	0	0	0	0	0	0	3910.2	4	476926	0	0	0	0	0	0	0	4125.2
ispd18_test7	10	869909	0	0	0	0	0	0	0	6723.5	16	818063	2	0	0	0	2	0	0	7107.2
ispd18_test8	11	900026	0	0	0	0	0	0	0	7014.7	22	846126	0	0	0	0	0	0	0	7513.9
ispd18_test9	4	862491	0	0	0	0	0	0	0	5978.5	11	817439	0	0	0	0	0	0	0	6389.5
ispd18_test10	64	930801	335	243	0	24	68	0	0	23757.1	64	883065	172	132	0	9	31	0	0	18499
Comp	1.29	1.058	-	-	-	-	-	-	-	0.9124	1.00	1.000	-	-	-	-	-	-	-	1.000

TABLE VI: The detailed routing results of the ISPD-2019 benchmarks for the DP-based method using Techniques (i)+(ii)+(iii), and COALA with obstacle awareness.

Case name	DP-based with Technique (i)+(ii)+(iii) [7]+[3]+[13]										COALA w/ Obstacle Awareness									
	Itr	IDV	DV	MS	CS	EL	PL	AS	MA	CPU	Itr	IDV	DV	MS	CS	EL	PL	AS	MA	CPU
ispd19_test1	8	6323	0	0	0	0	0	0	0	543.1	5	6091	0	0	0	0	0	0	0	590.4
ispd19_test2	9	190322	4	0	0	0	0	4	0	8122.92	11	178517	0	0	0	0	0	0	0	8871.5
ispd19_test3	45	18033	0	0	0	0	0	0	0	943.7	44	16631	0	0	0	0	0	0	0	968.0
ispd19_test4	-	191200	-	-	-	-	-	-	-	-	-	164202	-	-	-	-	-	-	-	-
ispd19_test5	-	35598	-	-	-	-	-	-	-	-	-	27810	-	-	-	-	-	-	-	-
ispd19_test6	22	288074	1	0	0	0	0	0	1	19948.4	22	275084	3	0	0	1	1	0	1	29188.0
ispd19_test7	18	670997	0	0	0	0	0	0	0	28524.3	21	618222	0	0	0	0	0	0	0	31779.9
ispd19_test8	18	2181157	0	0	0	0	0	0	0	30724.0	12	2085455	0	0	0	0	0	0	0	32168.5
ispd19_test9	27	3615564	0	0	0	0	0	0	0	47906.4	27	3481976	1	0	0	0	0	0	1	55338.3
ispd19_test10	64	3248972	21	2	6	6	7	0	0	53466.1	64	3012946	17	0	2	7	8	0	0	63044.8
Comp	1.09	1.092	-	-	-	-	-	-	-	0.882	1.00	1.000	-	-	-	-	-	-	-	1.000

violations, "MA" the minimum area violations, and "CPU" the running time in seconds. Table V gives the DRV results of the ISPD-2018 benchmarks by comparing the DP-based approach with all techniques and the obstacle-aware COALA framework. Similarly, Table VI gives the DRV results of the ISPD-2019 benchmarks. Due to some technical problems, ispd19_test4 and ispd19_test5 cannot be routed with TritonRoute. The binary can successfully complete region query, pin access, guide query, gr pin query, and track assignment, but it breaks down during the convergence of detailed routing with a connectivity error prompt. We have checked all connections in the two designs and failed to fix the problem.

In both ISPD-2018 benchmarks and ISPD-2019 benchmarks, TritonRoute reports 0 DRVs in the first 9 cases, but the DRC reports from Innouvs may find few missing DRVs. Thus, these DRVs could be regarded as resolvable once TritonRoute uses the same standards as those in Innovus. In ispd18_test10, our COALA framework reduces the total number of DRVs by almost 50%, which is achieved by reducing the violations of metal shorts, EOL spacings, and PRL spacings. In ispd19_test10, although the total numbers of DRVs are close, our COALA framework generates 0 metal short violation, which is the most indicative DRV type for congestion regions. Although the two compared approaches achieve similar results after detailed routing in most of the cases, the DP-based method suffers from 29%/9% higher detailed routing iterations and 5.8%/9.2% more initial DRVs averagely for the testcases of the ISPD-2018/2019 benchmarks. In addition, it can be observed that COALA can gets fewer initial DRVs in ispd19_test4 and ispd19_test5 as the other cases, which may imply that COALA also outperforms the DP-based approach for the two cases.

Figs. 17 and 18 show other statistics related to metal utilization in the detailed routing results of the DP-based approach with all techniques, which are normalized to those generated from the COALA framework. Fig. 17 shows the total metal and via utilizations in each testcase (T1 to T10 separately denote test1 to test10 in each benchmark set). Note that the data of T4 and T5 of ISPD-2019 benchmarks are absent. It can be observed that the metal utilizations in all cases are really close to each other, yet the DP-based layer assignment method averagely suffers from 1% more via counts in the large and congested cases such as ispd18_test5 to ispd18_test10 and ispd19_test8 to ispd19_test10. Fig. 18 shows the breakdown of the utilization ratio of the DP-based approach to the COALA framework for each via and metal layer averaged over the 10 testcases in each benchmark set. The bar graph shows close utilizations from M1 to M3 between the DP-based approach and the COALA framework. However, the DP-based approach results in much larger metal and via utilizations from M3 to M6 and much fewer utilizations from M7 to M9. The distribution of metal and via utilizations reveal that our COALA framework can generate more sparser routing results with careful via usage. Although the DP-based layer assignment achieves slightly shorter total metal length, it causes more vias in large and congested cases. The via overhead should be generated during the conventional rip-up and rerouting process. With the increase of iterations, the rip-up and rerouting paths also require more metal lengths and vias to reduce overflow. In addition, if the neighboring region is as crowded as an overflow region, the DP-based approach may easily generate DRVs in the detailed routing stage, such as the situation we can observe in ispd18_test10.

VI. Conclusions

This paper is the first work to solve the layer assignment problem by concurrently considering the wire segments of all nets. The presented concurrent layer assignment framework assigns the segments from the lowest to the highest available layers. Based on this concurrent scheme, the via demands can be fully considered and overflow can be maximally avoided

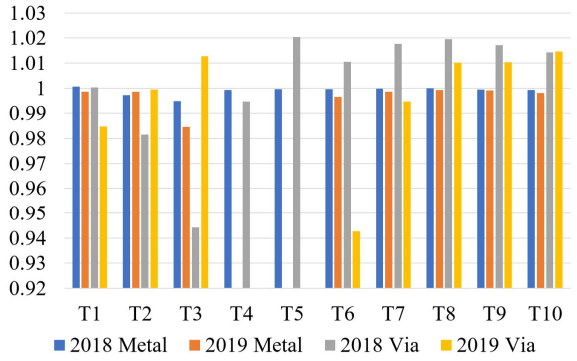

Fig. 17: Metal and via utilizaitons in the ISPD-18 and ISPD-19 benchmarks. The data are the ratios of the DP-based approach to the COALA framework.

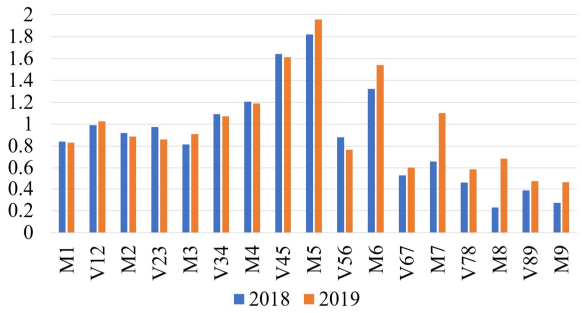

Fig. 18: The breakdown of metal and via utilizations for each via and metal layer. The data are the ratios of the DP-based approach to the COALA framework.

with the help of the proposed demand prediction map. The layer assignment procedure is majorly composed of three stages. First, complete segments are assigned until there are only scattered spaces left. Then, the scattered spaces are filled with fragmented segments to fully use routing resources. Last, the unassigned wire segments are re-routed without ripping up any segments to effectively reduce inevitable overflows caused by non-ideal 2D global routing results. In addition, we present the obstacle awareness strategy, which provides critical segments higher priorities to escape from obstacle area and forbids segment assignments that are expected to cause overflows. Both the experiments using the ISPD-2018 and ISPD-2019 benchmarks show that COALA can significantly reduce the maximum overflow and overflow tiles with less wirelength and runtime compared to an implemented DP-based layer assignment approach followed by a conventional rip-up and rerouting process. Additionally, the results of the ISPD-2019 benchmarks also show that the obstacle awareness strategy is necessary in the concurrent layer assignment scheme. Finally, the detailed routing results also verify that the COALA framework can produce sparser metal and via utilizaitons, which also contributes to fewer DRVs in the detailed routing stage.

REFERENCES

[1] M. Cho, K. Liu, K. Yuan, and D.Z. Pan, "BoxRouter 2.0: Architecture and implementation of a hybrid and robust global router," in *Proc. International Conference on Computer-Aided Design*, pp. 503–508, 2007.

[2] Y. Chang, Y. Lee, and T. Wang, "NTHU-Route 2.0: A fast and stable global router," in *Proc. International Conference on Computer-Aided Design*, pp. 338–343, 2008.

[3] K.-R. Dai, W.-H. Liu, and Y.-L. Li, "NCTU-GR: Efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing," *IEEE Transactions on Very Large Scale Integration Systems* , vol. 20, no. 3, pp. 459–472, 2012.

[4] C.-H. Hsu, H.-Y. Chen, and Y.-W. Chang, "Multilayer global routing with via and wire capacity considerations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 685–696, 2010.

[5] Y.-J. Jiang and S.-Y. Fang, "COALA: Concurrently assigning wire segments to layers for 2D global routing," in *Proc. International Conference on Computer-Aided Design*, pp. 1–8, 2020.

[6] A. B. Kahng, L. Wang and B.Xu, "TritonRoute-WXL: The Open Source Router with Integrated DRC Engine," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[7] T.-H. Lee and T.-C. Wang, "Congestion-constrained layer assignment for via minimization in global routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* , vol. 27, no. 9, pp. 1643–1656, 2008.

[8] T.-H. Lee and T.-C. Wang, "Robust layer assignment for via optimization in multi-layer global routing," in *Proc. International Symposium on Physical Design*, pp. 159–166, 2009.

[9] D. Liu, B. Yu, S. Chowdhury, and D. Z. Pan, "TILA-S: Timing-driven incremental layer assignment avoiding slew violations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 231–244, 2018.

[10] W.-H. Liu and Y.-L. Li, "Negotiation-based layer assignment for via count and via overflow minimization," in *Proc. Asia and South Pacific Design Automation Conference*, pp. 539–544, 2011.

[11] W.-H. Liu, Y.-L. Li, and C.-K. Koh, "A fast maze-free routing congestion estimator with hybrid unilateral monotonic routing," in *Proc. International Conference on Computer-Aided Design*, pp. 713–719, 2012.

[12] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 5, pp. 709–722, 2013.

[13] J. Liu, C. W. Pui, F. Wang, and E. F. Y. Young, "CUGR: Detailed-Routability-Driven 3D Global Routing with Probabilistic Resource Model," in *Proc. Design Automation Conference*, pp. 1–6, 2020.

[14] V. Livramento, D. Liu, S. Chowdhury, B. Yu, X. Xu, D. Z. Pan, J. L. Gntzel, and L. C. V. Santos, "Incremental layer assignment driven by an external signoff timing engine," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 7, pp. 1126–1139, 2017.

[15] M. D. Moffitt, "MaizeRouter: Engineering an effective global router," in *Proc. Asia and South Pacific Design Automation Conference*, pp. 226–231, 2008.

[16] M. M. Ozdal and M. D. F. Wong, "Archer: A history-driven global routing algorithm," in *Proc. International Conference on Computer-Aided Design*, pp. 488–495, 2007.

[17] Z. Qi, Y. Cai, Q. Zhou, Z. Li, and M. Chen, "VFGR: A very fast parallel global router with accurate congestion modeling," in *Proc. Asia and South Pacific Design Automation Conference*, pp. 525–530, 2014.

[18] J. A. Roy, and I. L. Marcov, "High-performance routing at the nanometer scale," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 6, pp. 1066–1077, 2007.

[19] H. Shojaei, A. Davoodi, and J. Linderoth, "Planning for local net congestion in global routing," in *Proc. International Symposium on Physical Design*, pp. 85–92, 2013.

[20] D. Shi, E. Tashjian, and A. Davoodi, "Dynamic planning of local congestion from varying-size vias for global routing layer assignment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 8, pp. 1301–1312, 2017.

[21] T. Wu, A. Davoodi, and J. T. Linderoth, "GRIP: Scalable 3D global routing using integer programming," in *Proc. Design Automation Conference*, pp. 320–325, 2009.

[22] Y. Xu and C. Chu, "MGR: Multi-level global router," in *Proc. International Conference on Computer-Aided Design*, pp. 250–255, 2011.

[23] Y. Zhang, Y. Xu, and C. Chu, "FastRoute3.0: A fast and high quality global router based on virtual capacity," in *Proc. International Conference on Computer-Aided Design*, pp. 344–349, 2008.

[24] S. Dolgov, A. Volkov, L. Wang, and B. Xu, "2019 CAD Contest: LEF/DEF Based Global Routing," in *Proc. International Conference on Computer-Aided Design*, pp. 1–4, 2019.

[25] S. Mantik, G. Posser, W. K. Chow, Y. Ding, and W. H. Liu, "ISPD 2018 Initial Detailed Routing Contest and Benchmarks," in *Proc. International Symposium on Physical Design*, pp. 140–143, 2018.

[26] W. H. Liu, S. Mantik, W. K. Chow, and Y. Ding, "ISPD 2019 Initial Detailed Routing Contest and Benchmark with Advanced Routing Rules," in *Proc. International Symposium on Physical Design*, pp. 147–151, 2019.

[27] The OpenROAD Project TritonRoute, https://github.com/The-OpenROAD-Project/TritonRoute

[28] Cadence Innovus Implementation System user guide, https://www.cadence.com/en_US/home.html